

Tharan Gregory Lanier (State Bar No. 138784)
 tglanier@JonesDay.com
 Nathaniel P. Garrett (State Bar No. 248211)
 ngarrett@JonesDay.com
 Joshua L. Fuchs (*Pro Hac Vice*)
 jlfuchs@JonesDay.com
 Joseph M. Beauchamp (*Pro Hac Vice*)
 jbeauchamp@jonesday.com
 JONES DAY
 555 California Street, 26th Floor
 San Francisco, CA 94104
 Telephone: +1.415.626.3939
 Facsimile: +1.415.875.5700

Kristin L. Cleveland (State Bar No. 184639)
 kristin.cleveland@klarquist.com
 John D. Vandenberg (*Pro Hac Vice*)
 john.vandenberg@klarquist.com
 J. Christopher Carraway (*Pro Hac Vice*)
 chris.carraway@klarquist.com
 Roy Chamcharas (*Pro Hac Vice*)
 roy.chamcharas@klarquist.com
 KLARQUIST SPARKMAN, LLP
 121 SW Salmon Street, Suite 1600
 Portland, OR 97204
 Telephone: +1.503.595.5300
 Facsimile: +1.503.595.5301

Attorneys for Defendants
 SAP SE, SAP AMERICA, INC., AND SAP
 LABS, LLC

Kenneth A. Gallo (*Pro Hac Vice*)
 kgallo@paulweiss.com
 David J. Ball (*Pro Hac Vice*)
 dball@paulweiss.com
 William B. Michael (*Pro Hac Vice*)
 wmichael@paulweiss.com
 Crystal M. Johnson (*Pro Hac Vice*)
 cjohnson@paulweiss.com
 PAUL, WEISS, RIFKIND, WHARTON &
 GARRISON LLP
 2001 K Street NW
 Washington, DC 20006-1047
 Telephone: +1.202.223.7356
 Facsimile: +1.202.204.7356

UNITED STATES DISTRICT COURT
 NORTHERN DISTRICT OF CALIFORNIA
 SAN FRANCISCO DIVISION

**TERADATA CORPORATION,
 TERADATA US, INC., and TERADATA
 OPERATIONS, INC.,**

Plaintiffs/Counterclaim-Defendants,
v.

SAP SE,
Defendant/Counterclaim-Plaintiff,
and

SAP AMERICA, INC., and SAP LABS,
LLC,
Defendants.

Case No. 3:18-CV-03670-WHO

**DEFENDANT SAP SE'S OPENING
 CLAIM CONSTRUCTION BRIEF**

Date: March 27, 2020
 Time: 10:00 a.m.
 Place: Ctrm 2, 17th floor
 Before: Hon. William H. Orrick

TABLE OF CONTENTS

		Page
1		
2		
3	LIST OF EXHIBITS	iv
4	I. INTRODUCTION	1
5	II. LEVEL OF ORDINARY SKILL IN THE ART	1
6	III. CLAIM TERMS TO BE CONSTRUED	2
7	A. '179 Patent - "Optimal Access Plan" / "Optimize"	2
8	1. '179 Patent Background.....	2
9	2. The Claims Do Not Require Finding The Absolute Best Plan	3
10	B. '179 Patent - "Query Optimization Graph"	4
11	C. '179 Patent - "Query Block"	7
12	D. '421 Patent - "Share A Consistent View Of Said Database Information"	8
13	1. '421 Patent Background.....	8
14	2. "Shar[ing] A Consistent View" Requires Updating Row-Format Data And Column-Format Data In The Same Database Transaction.....	10
15	E. '421 Patent - "Wherein Generating The Query Response Accesses Only One Or More Columns Needed Directly For Generating The Query Response"	11
16	F. '437 Patent - "A Plurality Of Data Dictionary Cache At An Application Level"	13
17	1. '437 Patent Background.....	13
18	2. A "Plurality Of Data Dictionary Cache At An Application Level" Does Not Require A Correspondence Between Application Servers And Data Dictionary Caches.....	14
19	G. '321 Patent - "Mapping" / "Mapping Table"	15
20	1. '321 Patent Background.....	16
21	2. The Claims' "Mapping" Is Stored	16
22	3. The Claims' "Mapping" Is Always Pairwise.....	17
23	H. '321 Patent - "Online Analytical Processing Cube"	18
24		
25		
26		
27		
28		

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28

I.	'321 Patent - “[Means For] Invoking An Online Analytical Processing Component To Fill The Online Analytical Processing Cubes With Transactional Data”	19
1.	There Is No Basis For Requiring That The OLAP Cube Be Empty.....	20
2.	The Specification Identifies An “Application” As The Associated Structure For Performing The “Invoking” Function	20
J.	'516 Patent - “Eviction Policy Plug-In” / “Storage Policy Plug-In” / “Plug-In”	21
1.	'516 Patent Background.....	22
2.	The Context Of The '516 Patent Controls The Meaning Of The “Plug-In Terms”	23
IV.	CONCLUSION.....	25

TABLE OF AUTHORITIES**Page(s)****Cases**

<i>Medtronic, Inc. v. Advanced Cardiovascular Systems, Inc.</i> , 248 F.3d 1303 (Fed. Cir. 2001)	20
<i>Phillips v. AWH Corp.</i> , 415 F.3d (Fed. Cir. 2005)	5
<i>Teva Pharms., Inc. v. Sandoz, Inc.</i> , 135 S. Ct. 831 (2015)	1
<i>Zeroclick, LLC v. Apple Inc.</i> , 891 F.3d 1003 (Fed. Cir. 2018)	21

Statutes

35 U.S.C. 112	19, 20, 21
---------------------	------------

LIST OF EXHIBITS

'179 Patent

- Exhibit 1 “**'179 Pat.**”—U.S. Patent No. 7,617,179
- Exhibit 2 “**'179 Pros. Hist.**”—Prosecution History for U.S. Patent No. 7,617,179 (excerpts)
- Exhibit 3 “**Ramakrishnan**”—Ramakrishnan, et al. “Database Management Systems,” McGraw Hill Press (third ed. 2003) (excerpts)
- Exhibit 4 “**Bowman**”—Bowman, et al., “Join Enumeration in a Memory-Constrained Environment,” Proceedings, 16th IEEE Data Engineering Conference, San Diego, California (Mar. 2000)
- Exhibit 5 “**Elmasri**”—R. Elmasri and S. Navathe, “Fundamentals of Database Systems,” (Benjamin/Cummings Pub. Co., 1989) (excerpts)
- Exhibit 6 “**Paulley**”—U.S. Patent App. Pub. No. 2002/0116357

'421 Patent

- Exhibit 7 “**'421 Pat.**”—U.S. Pat No. 9,626,421
- Exhibit 8 “**'421 Pros. Hist.**”—Prosecution History for U.S. Pat No. 9,626,421 (excerpts)
- Exhibit 9 “**'893 App.**”—U.S. Prov. Pat. App. No. 60/994,893

'437 Patent

- Exhibit 10 “**'437 Pat.**”—U.S. Pat. No. 7,421,437
- Exhibit 11 “**'437 Pros. Hist.**”—Prosecution History for U.S. Pat. No. 7,421,437

'321 Patent

- Exhibit 12 “**'321 Pat.**”—U.S. Pat. No. 8,214,321
- Exhibit 13 “**'321 Pros. Hist.**”—Prosecution History for U.S. Pat. No. 8,214,321 (excerpts)
- Exhibit 14 “**MS Comp. Dict.**”—Microsoft Computer Dictionary, 5th ed. (excerpts)

'516 Patent

- Exhibit 15 “**'516 Pat.**”—U.S. Pat. No. 7,437,516
- Exhibit 16 “**Dict. of Computer and Internet Terms**”—Dictionary of Computer and Internet Terms, Barron’s Educational Services, Inc. (1998) (excerpts)
- Exhibit 17 “**Modern Dict. of Elects.**”—Modern Dictionary of Electronics, Seventh Edition, Revised and Updated, Rudolf F. Graf, Newness (1999) (excerpts)

1 Exhibit 18 “**Decasper**”—D. Decasper, et al., “Router Plugins: A Software Architecture for
2 Next Generation Routers,” Proceedings of the ACM SIGCOMM '98 Conference on
3 Applications, Technologies, Architectures, and Protocols for Computer
4 Communication, 1998, Vancouver, British Columbia, Canada, 1998.
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28

1 **I. INTRODUCTION**

2 Patents are written for skilled artisans and interpreted through their eyes. The primary
 3 evidence supporting interpretation of claim terms is the intrinsic evidence, namely the patent and
 4 its prosecution history. But, where the intrinsic evidence does not fully resolve a dispute, the best
 5 evidence of how a skilled artisan would understand a patent’s intrinsic disclosures and
 6 terminology is the analysis of an expert in the field familiar with skilled artisans and the
 7 technical terms they use, as well as how the technology works. Thus, while patent claim
 8 construction is a question of law, it often depends on subsidiary questions of fact on which an
 9 expert may opine, including “the background science or the meaning of a term in the relevant art
 10 during the relevant time period.” *Teva Pharms., Inc. v. Sandoz, Inc.*, 135 S. Ct. 831, 841 (2015).

11 Here, expert testimony will be helpful to the Court in resolving the parties’ disputes on
 12 claim construction. SAP provides that evidence in the accompanying Declaration of Dr. David
 13 Maier, after complying with its PLR 4-2 and 4-3 obligations to identify its technical expert
 14 witness, describe his opinions and identify any requested factual findings. Teradata did neither,
 15 nor did it depose Dr. Maier. By Teradata’s choice, expert testimony in this matter, and proposed
 16 findings of fact based thereon, is provided by SAP alone.

17 **II. LEVEL OF ORDINARY SKILL IN THE ART**

18 SAP’s expert, Dr. David Maier, identifies the qualifications of a person of ordinary skill
 19 in the art (“POSITA”) for each of the five asserted patents. (*See* accompanying Declaration of
 20 David Maier In Support Of SAP’s Markman Brief (“Maier”), at ¶¶ 27–42; *see also* JCCS, at 56–
 21 63.) Dr. Maier also explains how a POSITA would have understood the terms at issue. Teradata
 22 has proposed no findings of fact on the level of skill in the art. (*See* JCCS, at 64.) Teradata does
 23 not object to SAP’s proposed qualifications of a POSITA for four of the asserted patents (the
 24 ’179 patent, the ’421 patent, the ’437 patent, and the ’516 patent). For the ’321 patent, Teradata
 25 offers, without support, an alternative proposal for the qualifications of a POSITA. That
 26 alternative, even if accepted, does not affect Dr. Maier’s claim construction opinions. (Maier, ¶
 27 39.)

III. CLAIM TERMS TO BE CONSTRUED

SAP briefs only the ten terms identified in the JCCS as most significant. Teradata has indicated that it will do the same.

A. '179 Patent - "Optimal Access Plan" / "Optimize"

SAP	Teradata
"the estimated best plan, considering estimates of execution costs, among the plans considered"	"the access plan that has the lowest execution cost associated with it"
"find/found the estimated best plan, considering estimates of execution costs, among the plans considered"	"find the lowest execution cost"

Dispute: Do the claims require finding the absolutely best, or "lowest cost," plan?

1. '179 Patent Background¹

The '179 patent relates to "query optimization" in a relational database system. Such systems store potentially vast amounts of data organized into one or more tables. (Maier, ¶ 44.) This data is accessed by executing a query that requests or updates specific information found in the system. (*Id.*) Often, this access requires gathering and combining information from several different tables. The task of a "query optimizer" is to identify one plan for performing these access steps from among multiple possible plans. (*Id.*) To identify this plan the optimizer will determine a number of possible "access methods," or ways to obtain the needed data from various tables, *e.g.*, by selecting a particular index to access the desired data. In addition, the information from various tables might need to be combined, or "joined." (*Id.*) Typically, only two tables can be joined at a time (either of which can be an intermediate table produced by joining two other tables, which can in turn be intermediate tables), so the optimizer will consider possible "join orders" or sequences in which to combine the data from the tables. (*Id.*) In

¹ Background sections are provided in this brief as an introduction to the patents. The citations in these Background sections are not provided to support any particular claim construction and, in some instances, were not set forth in the JCCS.

1 addition, the optimizer will identify possible “join methods,” or ways to combine the data from
 2 the tables. (*Id.*) The optimizer also estimates the “execution costs”² associated with each option
 3 considered. (*Id.*)

4 After considering the estimated execution cost for various options for various parts of the
 5 plan, the optimizer will select a single combination of plan steps for executing the query—an
 6 “optimal access plan.” (*Id.*, ¶ 45.) Depending on the complexity of the query and the database
 7 structure, the number of possible plans could be extremely large, and the optimizer might
 8 consider only a subset of them. (*Id.*)

9 **2. The Claims Do Not Require Finding The Absolute Best Plan**

10 To determine the absolute best access plan among myriad candidates would require
 11 exhaustive comparisons of *all* possible candidates. But the ’179 patent does not demand such
 12 theoretical perfection from a query optimizer. Rather, as it explains, “[t]he role of a query
 13 optimizer in a relational DBMS system is to find an *adequate* execution plan from a search space
 14 of *many* semantically equivalent alternatives.” (Ex. 1³, ’179 Pat., at 3:1–4 (emphasis added).)

15 Given the practical constraints, optimizers typically do not consider all possible plans and
 16 do not necessarily find the absolute best plan. This pragmatic approach was widely recognized
 17 by those in the art, as reflected in the literature:

18 The central task of an optimizer is to find a good plan for evaluating such
 19 expressions. . . . Typically, an optimizer considers a subset of all possible plans
 because the number of possible plans is very large.

20 (Ex. 3, Ramakrishnan, at p. SAP_04710096, (cited during the prosecution of the ’179 patent,
 21 hence, intrinsic evidence).)

22 A query optimizer often uses heuristics to simplify the search process:

23 However, producing an optimal access plan for an arbitrary SQL query is an NP-
 24 complete problem . . . ; to discover an optimal strategy requires an exhaustive

25 ² The parties have agreed to the construction of “execution cost” as “the estimated amount of
 26 computer system resources associated with performing a task.” (ECF 206 (JCCS), p. 1.)

27 ³ All Exhibits cited in this brief are identified in the List of Exhibits, *supra* at iv-v, and are
 28 attached to the accompanying Declaration of Marla Beier. Citations to patents are in the form of
 column:line numbers. Other citations are to Bates Nos.

search. Consequently, optimizers often use heuristics . . . to reduce the number of strategies that the plan selection phase must consider.

(Ex. 4, Bowman, at p. SAP_04708455. (citations omitted) (intrinsic evidence).)

The term *optimization* is actually a misnomer because in some cases the chosen execution plan is not the optimal (best) strategy—it is just a *reasonably efficient strategy* for executing the query. Finding the optimal strategy is usually too time-consuming except for the simplest of queries and may require information on how the files are implemented and even on the contents of the files—information that may not be fully available in the DBMS catalog.

(Ex. 5, Elmasri, at p. SAP_04709014 (emphasis is original) (extrinsic evidence). *See also* JCCS, at 57:3–8 (proposed FOF); Maier, ¶¶ 61–63 (identifying and explaining similar statements from both the intrinsic and extrinsic evidence).)

Thus, the claim terms “optimal access plan” and “optimize” refer to finding an estimated best plan *among those plans considered*, as urged by SAP. (Maier, ¶ 64.) These terms do not require considering every possible plan and selecting the absolute best plan (*id.*), as implied by Teradata’s proposed construction.

B. ’179 Patent - “Query Optimization Graph”

SAP	Teradata
“an internal representation of a query block or derived table block, structured as a graph”	“a representation of a query block in the form of a hypergraph (a graph in which an edge can join any number of vertices) that includes vertices (nodes) and edges (connections between two nodes), including hyperedges (a connection that can link more than two nodes) in which nodes can represent subplans and quantifiers; and in which each node representing subplans includes an array of access methods and an array of join methods”

Dispute: Should the preferred embodiment of a query optimization graph set forth in the specification, *i.e.*, a “hypergraph,” be read into the claims?

For a query optimizer to analyze a query, it is necessary for the optimizer to create some type of internal representation of that query. (Maier, ¶ 65.) The ’179 patent refers to this internal representation of the query as a “query optimization graph,” or “QOG.” (*See, e.g.*, Ex. 1, ’179 Pat., at 12:3–7.) In particular, the ’179 patent discloses that a “QOG,” as “the *internal*

1 *representation of a query block of Derived Table Block (DTB),*” serves as “the input to the plan
2 generation phase of the query optimizer.” (*Id.* (emphasis added).)

3 A POSITA would have recognized that a variety of internal representations for a query
4 were available in the art. (JCCS, at 56:25–57:1 (proposed FOF); Maier, ¶ 66.) One category of
5 such representations was “graphs,” and based on the express language of the claim term “query
6 optimization graph” a POSITA would have recognized that the ’179 patent contemplated the use
7 of a “graph” as an internal representation for the query. (*Id.*)

8 The preferred embodiment of the ’179 patent describes a particular type of graph, a
9 “hypergraph” (*id.*, ¶ 67.), and Teradata seeks to construe “query optimization graph” to require a
10 hypergraph. However, the claims themselves demonstrate that the term QOG is not limited to
11 this preferred embodiment, and does not require a hypergraph. Claim 3 depends from claim 1
12 and adds the further requirements that the QOG of claim 1 includes “plan nodes representing
13 subplans and quantifiers of each query block.” (Ex. 1, ’179 Pat., at 39:11–14.) Teradata’s
14 proposed construction requires the QOG (of claim 1) to include such details from claim 3: a
15 “hypergraph . . . in which nodes can represent subplans and quantifiers.” If this is required by
16 claim 1’s use of the term QOG, then claim 3 is rendered superfluous. (Maier, ¶ 75.) There is a
17 presumption against such a construction. *Phillips v. AWH Corp.*, 415 F.3d 1315 (Fed. Cir. 2005)
18 (en banc) (“[T]he presence of a dependent claim that adds a particular limitation gives rise to a
19 presumption that the limitation in question is not present in the independent claim.”).

20 That a QOG is not limited to hypergraphs is further evidenced by additional intrinsic
21 evidence including the cited prior art, the Examiner’s statements, and the inventor’s arguments.
22 The Patent Office Examiner recognized that a QOG need not be a hypergraph. The Examiner
23 rejected the original claims of the ’179 patent first over Ramakrishnan and then over Young-Lai.
24 The Examiner stated, “Ramakrishnan discloses . . . building a query optimization graph”
25 (Ex. 2, ’179 Pros. Hist., at SAP_02682791 (reproduced in JCCS Ex 1, at 3–4⁴.) *See also id.*, at
26 SAP_02682648–49.) Ramakrishnan shows a tree structure, which is a type of graph, but not a
27

28 ⁴ The JCCS erroneously cites to page 275 for this quotation. It actually appears on page 274.

1 hypergraph. (Maier, ¶ 68.) Later, the Examiner cited Young-Lai as showing a QOG. (Ex. 2, '179
 2 Pros. Hist., at SAP_02682594 (reproduced in JCCS Ex 1, at 3).) Young-Lai states: “[T]he input
 3 to the plan generation phase of the query optimizer is a Query Optimization Graph (QOG),
 4 pronounced ‘cog’. A QOG is the internal representation of a complete SQL statement, possibly
 5 composed of multiple ‘subquery blocks.’” (*Id.* (quoting Young-Lai).) Again, Young-Lai does not
 6 show a hypergraph. (Maier, ¶ 69.) Based on these rejections a POSITA would have understood
 7 that the Examiner interpreted “query optimization graph” not to be limited to the particular
 8 parameters used for the disclosed hypergraphs. (*Id.*, ¶ 71.)

9 The inventor’s response to the Examiner’s rejections further shows that a hypergraph is
 10 not required by the claims. (*See* Ex. 2, '179 Pros. Hist., at SAP_02682773-78, SAP_02682707-
 11 16 and SAP_02682626-32.) Rather than distinguish the claims by arguing that neither
 12 Ramakrishnan nor Young-Lai showed a hypergraph, the inventor successfully argued other
 13 aspects of the claims to gain their allowance. (Maier, ¶ 72.)

14 The extrinsic evidence also recognizes that a QOG need not be a hypergraph. For
 15 example, Paulley states that “[a] QOG [i.e. query optimization graph] is the internal
 16 representation of a complete SQL statement, possibly composed of multiple ‘subquery blocks.’”
 17 (Ex. 6, Paulley, at p. SAP_04710975, ¶ [0050].) Paulley describes a QOG having a graph
 18 structure, but not a hypergraph structure. (Maier, ¶ 70.)

19 Teradata likely will argue that the '179 patent provides a “definition” that expressly
 20 requires a QOG to be a hypergraph. Describing the particular parameters used for the QOG of
 21 the *preferred* embodiment, the '179 patent states: “Definition 4: A QOG is a representation of a
 22 single Derived Table Block (DTB). A QOG is defined as a hypergraph $Q = \langle V, E \rangle$ where V and
 23 E are defined as follows:” (Ex. 1, '179 Pat., at 14:47–49.) This explanation of the preferred
 24 embodiment is, understandably, not found in the Glossary before the “Summary of the
 25 Invention.” Rather, it is found in the “Detailed Description of a Preferred Embodiment” section
 26 of the patent. Further, “Definition 4” extends for approximately 95 additional lines, providing
 27 extensive explanation about an example QOG. Tellingly, Teradata cherry picks small parts of
 28

1 this section for its proposed construction, while the length and nature of this section of the '179
 2 patent (as well as its heading and content) clearly indicate that the section is an explanation of
 3 specific parameters relevant to a preferred embodiment and is not intended as a definition
 4 limiting the claims. (Maier, ¶ 73.)

5 This lack of intent to limit the claims by this embodiment's "Definition 4" is further
 6 supported by the context of the '179 specification. A review of the Detailed Description section
 7 of the '179 patent reveals a subsection entitled "C. Definition of Terms" that extends from the
 8 bottom of column 13 to the middle of column 21. This section is composed of lengthy
 9 explanatory "Definitions," in a style similar to "Definition 4." These "Definitions" contain
 10 detailed descriptions, pseudo code and examples that illustrate a preferred embodiment. In
 11 context, it is clear that these "Definitions" are not intended to be read into the claims. (*Id.*, ¶ 74.)

12 Teradata's reliance on the "Definition 4" QOG example is undermined not only by its
 13 willingness to cherry pick small portions of "Definition 4," but also by its willingness to adopt or
 14 propose different constructions for other terms provided with embodiment-specific "Definitions"
 15 (e.g., "quantifier" (*cf.*, Definition 3 of the '179 specification with agreed construction (JCCS at
 16 1)); "plan node" (*cf.* Definition 9 with Teradata's proposed construction (JCCS at 6)); "access
 17 methods" (*cf.* Definition 11 with agreed construction (JCCS at 1)); and "access plan" (*cf.*
 18 Definition 12 with Teradata's proposed construction (JCCS at 7)). It is inconsistent for Teradata
 19 to argue that a small portion of "Definition 4" is intended to limit the claims but other similarly
 20 labeled and presented "Definitions" are not.

21 **C. '179 Patent - "Query Block"**

SAP	Teradata
"an atomic portion of a query that can be separately optimized"	"a part of a query that contains multiple parts because the query contains derived tables, views, and/or subqueries"

25 Dispute: Is a "query block" the smallest part of a query that can be separately optimized?

26 SAP's proposed construction is derived from the Glossary of the '179 patent (placed
 27 before the "Summary of the Invention") which states: "A query block refers to an atomic portion
 28 or block of a query that has more than one block because the query contains derived tables,

views, and/or subqueries.” (Ex. 1, ’179 pat., at 4:43–45.) The word “atomic” reflects that the block cannot be broken into smaller portions that can be separately optimized. (Maier, ¶ 78.) SAP’s construction omits the second portion of this statement not because it is inaccurate, but because it is unnecessary, difficult for a jury to understand, and likely would lead to confusion. That is, a jury needs to be instructed as to what a query block is, but need not know why something is a query block.

Teradata’s proposed construction focuses on “why” something is a query block, but omits the clearest statement about what a query block is, *i.e.*, a smallest optimizable unit of a query. (Maier, ¶ 79.) Because of this omission Teradata’s proposal might be understood to encompass *any* “part” of a query that “contains derived tables, views, and/or subqueries,” even if that part is: 1) too small to be optimized; or 2) so large that it contains smaller portions that can be separately optimized. (*Id.*) Both of these outcomes would be contrary to the meaning of “query block” as set forth in the ’179 patent and understood by a POSITA. (*Id.*) Moreover, both of these outcomes would be inconsistent with the preferred embodiments of the ’179 patent, which build plans based on the smallest optimizable parts of a query—nothing smaller and nothing larger. (*Id.*)

D. ’421 Patent - “Share A Consistent View Of Said Database Information”

SAP	Teradata
“update the row-format data (by the relational database management system component) and update the column-format data (by the column-oriented data processing component) within the same database transaction”	“database information that if accessed by the relational database management component is the same as if accessed by the column-oriented data processing component”

Dispute: Does “share a consistent view of said database information” require that row-format data and column-format data be updated within the same database transaction?

1. ’421 Patent Background

Transactional data, such as records for particular sales, is typically entered or stored in a relational database in rows in a two-dimensional table. (Maier, ¶ 46) The table also has columns of values for attributes of the transaction. For example, there might be columns for the sales price, quantity sold, date and customer. (*Id.*) For updates to records, it is often more efficient to

1 store the data on a row-by-row basis – in a row format – for the respective records. (*Id.*) On the
 2 other hand, when analyzing such data, for example, to identify sales trends or find the average
 3 sales price, it is often more efficient to work with the data stored on a column-by-column basis –
 4 in a column format. (*Id.*) The '421 patent relates to a system which allows both row-format
 5 storage of database transactions (transactions represent, *e.g.*, requests to add, edit or delete a
 6 record) and column-format processing (for analyzing and aggregating data). (*Id.*)

7 In the '421 patent, the row-format data is stored and updated by a “relational database
 8 management system component,” and the column-format data is stored and updated by a
 9 “column-oriented data processing component.” (*See, e.g.*, Ex. 7, '421 Pat., at 2:28–40; Maier, ¶
 10 47.) The '421 patent describes how these systems are synchronized so that both are always
 11 working with exactly the same data as new transactions are entered into the database. (*See, e.g.*,
 12 Ex. 7, '421 Pat., at 9:26–40; Maier, ¶ 47.)

13 The specification of the '421 patent identifies SAP's MaxDB software as an example of a
 14 relational database management system for row-format data. (*See, e.g.*, Ex. 7, '421 Pat., at 9:14–
 15 16; Maier, ¶ 48.) Thus, MaxDB, or a similar relational database management system, handles
 16 transactions to update row-format data. (Maier, ¶ 47.) In handling transactions to update data,
 17 MaxDB supports the ACID properties (Atomicity, Consistency, Isolation, Durability) for
 18 database transactions. The point of these properties is to guarantee the validity of data even in the
 19 event of errors, concurrent accesses, power failures, etc. (Maier ¶ 48.)

20 The specification of the '421 patent identifies SAP's TREX software as an example of a
 21 column-oriented data processing system. (*See, e.g.*, Ex. 7, '421 Pat., at 9:16–18; Maier, ¶ 49.)
 22 TREX is optimized for read access, so requests to retrieve data are handled by TREX, which
 23 accesses the column-format data. (Maier, ¶ 49.)

24 When row-format data is updated, corresponding column-format data is also updated.
 25 (*Id.*) Further, the updates are performed such that the relational database management system
 26 component and the column-oriented data processing component “share a consistent view of said
 27
 28

1 database information.” (*Id.*) This ensures that the two components are working with the same
 2 data at all times. (*Id.*)

3 **2. “Shar[ing] A Consistent View” Requires Updating Row-Format**
 4 **Data And Column-Format Data In The Same Database Transaction**

5 The ’421 patent makes clear that column-format data and row-format data are updated
 6 within the same database transaction. (Maier, ¶ 81.) After a component of MaxDB (an example
 7 of a relational database management system) updates row-format data, “TREX 206 is notified
 8 about the changes and can update its own data with the help of the queue tables 216. *This*
 9 *happens within the same database transaction.* Accordingly, TREX 206 and MaxDB 204 share a
 10 consistent view of data.” (Ex. 7, ’421 Pat., at 9:35–40 (emphasis added); *see also* Ex. 9, ’893
 11 App., at pp. 12-13, 22.) Thus, the patent expressly explains to a POSITA that “share a consistent
 12 view of said database information” means column-format data and row-format data are updated
 13 “within the same database transaction.” (Maier, ¶ 83; *see also* JCCS, at 60:13–15 (proposed
 14 FOF).) This meaning was confirmed during prosecution by the inventors’ citation to this part of
 15 the specification to explain how MaxDB and TREX share a consistent view of data when
 16 updates are performed:

17 Accordingly, when the database information is updated, this first occurs in the
 18 row store. To ensure consistency between row store and column store, such that
 19 queries which are based on the column store always return a correct result, the
 20 column store is notified of the update to the row store and subsequently updates
 21 the database information in the column store in the same way. This is explained in
 detail in the specification at [0063], where it is emphasized in the last sentence
 that TREX 206 (column store) and MaxDB 204 (row store) share a consistent
 view of data.

22 (Ex. 8, ’421 Pros. Hist., at pp. SAP_02684072-73 (referring to paragraph [0063] of the published
 23 application, which includes the “same database transaction” language). (Maier, ¶ 84.)

24 Teradata’s proposed construction is insufficient because it fails to clearly state that row-
 25 format data and column-format data are updated in the same database transaction. If row-format
 26 data were to be updated in a first transaction, and column-format data were to be updated in a
 27 later, second transaction, the column-format data would potentially be different than the row-
 28

format data for somebody querying the data. In that case, the two components would not “share a consistent view” of said database information, which is contrary to the manner that updates are performed according to the ’421 patent.

E. ’421 Patent - “Wherein Generating The Query Response Accesses Only One Or More Columns Needed Directly For Generating The Query Response”

SAP	Teradata
“wherein generating the query response reads or writes data in only the column(s) needed to generate the query response” ⁵	“wherein generating the query response reads or writes data in only the column(s) consisting of the results to the query”

Dispute: Does the phrase “wherein generating the query response accesses only one or more columns needed directly for generating the query response” have its plain meaning?

This phrase is not a term of art, should be given its plain meaning, and needs no special construction. (Maier, ¶ 85–86; *see* JCCS, at 62:4–6 (proposed FOF).) As such, SAP proposes a construction that largely restates the phrase but makes it easier for a jury to understand, without changing its plain meaning. In contrast, Teradata’s proposed construction replaces “columns needed directly for generating the query response” with “column(s) consisting of the results of the query.” This replacement changes the plain meaning of the phrase and is contrary to the intrinsic evidence. (Teradata cites no extrinsic evidence in support of its proposed construction.)

In the specification of the ’421 patent, a column-oriented data processing component (e.g., in TREX) is used to access column-format data to retrieve the database information. The ’421 patent explains that, when generating a query response, a column-oriented data processing system accesses only the column(s) needed to generate the query response.

Column-oriented storage uses the observation that not all columns of a table are usually queried in a report or are needed to create the result. Compared to the relational model used in databases where all columns of two tables even those that are not necessary for the result are accessed the column-oriented approach yields a more lightweight solution. *Only the columns needed directly for creating⁶ the result have to be accessed.*

⁵ SAP has modified this construction from that appearing in the JCCS to correct a typographical error, *i.e.*, to remove the word “the” following the word “wherein.” (*Cf.* JCCS, p. 40, term 6.)

⁶ The term “creating” used here in the specification is synonymous with the term “generating” used in the claim.

(Ex. 7, '421 Pat., at 11:56–63 (emphasis added).) (In contrast, when accessing data by rows, it is necessary to access all columns within the rows, even if only one column or a few of the columns contains data responsive to the query, which may be much less efficient than accessing only the columns needed to respond to the query. (Maier, ¶ 86.))

During prosecution, the inventors also used the claim language “generating” when distinguishing the prior art:

Since Bourbonnais does not disclose a system using column storage, Bourbonnais also does not disclose that “wherein in response to a query request, said column-oriented data processing component generates a query response based on said database information stored in said column format, wherein generating the query response accesses only one of more *columns needed directly for generating the query response*”, as recited in claim 1 (as amended).

(Ex. 8, '421 Pros. Hist., at p. SAP_02683934 (cited by Teradata, JCCS Ex 3 at 97) (emphasis added).)

In contrast, according to Teradata’s proposed construction, a query response is generated by accessing data “in only the column(s) *consisting of the results* to the query” (emphasis added). However, Teradata’s language “the column(s) consisting of the results to the query” has a very different meaning than the claim language “columns needed directly for generating the query response.” In particular, Teradata’s proposed construction requires column(s) that store the results to a query, as opposed to the claim language, which requires columns that store the data needed to generate the results to a query. (Maier, ¶ 89.)

A POSITA would have recognized this difference and known that a column can store data needed to generate a query result without actually storing the result. (Maier, ¶¶ 87-88.) This can be illustrated by a simple hypothetical query. Given a table of employees that contains columns for the employee name, ID number, hire date and salary, a possible query could be “what is the average employee salary for all employees hired before 2015?” To generate a response to this query would require access to the “hire date” column and the “salary” column. From the information in these two columns the system could determine which employees were hired before 2015 and what their salaries are. The system could then calculate the average of those salaries and present the query result. The query result, however, *would not consist of* any of

the data from the “hire date” column, even though this information was directly necessary to obtain the result. Similarly, it would only be coincidental if any of the actual salaries found in the “salary” column were equal to the average salary computed in response to the query. (That is, the query result likely *would not consist of* any of the data from the “salary” column.) However, accessing the salary column was directly necessary to obtaining the query result. (*Id.*)

Teradata’s proposed construction would exclude the example above. (*Id.*, ¶ 89.) However, nothing in the ’421 patent supports such a construction or such an exclusion.

**F. ’437 Patent - “A Plurality Of
Data Dictionary Cache At An Application Level”**

SAP	Teradata
“multiple data dictionary caches at a layer of software that provides services to a user of data dictionary information, and obtains that data dictionary information from a data access layer”	“multiple data dictionary caches that are in corresponding application servers existing in a layer different from the data access level”

Dispute: Does the term “a plurality of data dictionary cache at an application level” require multiple data dictionary caches in corresponding application servers?

1. ’437 Patent Background

The ’437 patent relates to the management of data dictionaries in distributed systems. (Maier, ¶ 50.) A data dictionary stores information about the database. (*Id.*) For example, a data dictionary stores data type information, *i.e.*, “the semantic and syntactic properties of operational data, such as the type, length, and relationships of operational data.” (Ex. 10, ’437 Pat., at 1:38–40; Maier, ¶ 50.) Data types are specified for data objects stored in the system. For example, one data type might be “character” and another “numerical.” (Maier, ¶ 50.) When the system seeks to access or store a particular data object, it accesses the data dictionary to learn the characteristics of the data type for that object. (*Id.*)

A distributed system includes different computers connected over one or more networks. (*Id.*, ¶ 51.) The ’437 patent’s distributed system includes three layers with different roles in the management of data dictionary information: user layer, application layer, and data access layer. (*Id.*) In general, the user layer provides an interface between the distributed system and users.

1 The data access layer provides a persistent data store for the distributed system, storing data
 2 dictionary information in one or more data dictionaries. (*Id.*) The application layer is situated
 3 between the user layer and the data access layer, providing services to the user layer and, as
 4 needed, obtaining information from the data access layer. (*Id.*) The application layer can include
 5 application servers. (*Id.*) An application server is “a computing device that hosts application-
 6 level processing for application software.” (*See* JCCS, p. 3 (’437 Agreed Constructions).)

7 In the ’437 patent, one or more data dictionary caches are provided within the application
 8 layer for storing frequently-used information from the data dictionary. (Maier, ¶ 52; Ex. 10, ’437
 9 Pat., at 3:44–47.) Software in the application layer may receive from the user layer a request for
 10 data type information. (Maier, ¶ 52.) If one of the application layer’s data dictionary caches
 11 includes the requested data type information, that information is (quickly) accessed in the data
 12 dictionary cache and provided back to the user layer. (*Id.*) Otherwise, the requested data type
 13 information is (more slowly) obtained from a data dictionary at the data access layer. (*Id.*)

14 **2. A “Plurality Of Data Dictionary Cache At An**
 15 **Application Level” Does Not Require A Correspondence**
 16 **Between Application Servers And Data Dictionary Caches**

17 SAP’s proposed construction for “a plurality of data dictionary cache at an application
 18 level” makes the phrase easier for a jury to understand. In the ’437 patent, the application level is
 19 a layer of software, situated between the user layer and the data access layer, providing services
 20 to the user layer and, as needed, obtaining information from the data access layer. (Ex. 10, ’437
 21 Pat., at 1:48–55; Maier, ¶ 91.) In the context of the data dictionary caches in the claims, the
 22 application level (1) is a layer of software that (2) provides services to a user of data dictionary
 23 information and (3) obtains data dictionary information from a data access layer. The ’437 patent
 24 clearly explains to a POSITA that “a plurality of data dictionary cache at an application level”
 25 includes such features. (Maier, ¶ 91; see also JCCS, at 54:23–27 (proposed FOF).)

26 In contrast, Teradata’s proposed construction includes language – “in corresponding
 27 application servers” – that does not make the claim phrase easier to understand. In the ’421
 28 patent, an “application server” is “a computing device that hosts application-level processing for

application software.” (See JCCS, p. 3 (’437 Agreed Constructions).) The term data dictionary cache “refers to memory (or a region of memory) in an application server that stores information from an associated data dictionary.” (Ex. 10, ’437 Pat., at 3:55–57.) In other words, combining these definitions, a data dictionary cache is memory (or a region of memory) in a computing device that hosts application-level processing for application software. (Maier, ¶ 92.) Even if *arguendo* some examples in the ’437 patent show multiple data dictionary caches in an application layer along with application servers (*see, e.g.*, the data dictionary caches 330, 340 and application servers 222–224 in FIG. 3 of the patent), no 1:1 correspondence or other correspondence between data dictionary caches and application servers is required, as Teradata’s proposed construction implies. (*Id.*)

The prosecution history confirms that no correspondence between data dictionary caches and application servers at application level is required. (*Id.*, ¶ 93.) During prosecution, the Examiner repeatedly cited U.S. Patent No. 5,596,744 to Dao as showing the “application level” language of the claims. In particular, the Examiner wrote, “Dao clearly suggests the aforementioned features, particularly the application level and the data access level, in Figure 2: the data dictionary cache is located at the DIM (application level), and the data dictionary is located at the SDD server (data access level).” (Ex. 11, ’437 Pros. Hist., at pp. SAP_02681910, SAP_02681929.) FIG. 2 of Dao shows an SDD cache that caches data dictionary information from an SDD server and provides it to an optimizer. Dao does not show any correspondence between data dictionary caches and application servers (Maier, ¶ 93), as Teradata urges for its construction. The inventors eventually distinguished Dao on other grounds.

G. ’321 Patent - “Mapping” / “Mapping Table”

SAP	Teradata
“creating and persistently storing an association between two data elements in a computer system such that a computer can locate a data element using that association”	“associating or assigning”
“a computer-implemented data structure that holds associations or assignments, each between two data elements in a computer system”	“a computer-implemented data structure that holds associations or assignments”

1 Disputes: Is a mapping stored? Is a mapping between two data elements?

2 1. **'321 Patent Background**

3 Data often is input into a database in response to some type of transaction. (Maier, ¶ 53.)
 4 For example, when a sale is made, or a person is hired, data representing the transaction is input
 5 into the database. (*Id.*) Typically, transactional data is entered and stored in two-dimensional
 6 tables, which can be visualized as a two-dimensional data structure of rows and columns, like a
 7 simple spreadsheet. (*Id.*) This type of processing in a database is often referred to as online
 8 transaction processing, or “OLTP.” (*Id.*)

9 Once entered into a database, transactional data can be searched, grouped, and otherwise
 10 analyzed in order to provide insights and information to the database user. (*Id.*, ¶ 54.) For
 11 example, transactional data can be analyzed to identify sales trends, overhead, and the like. (*Id.*)
 12 Such analysis, often referred to as online analytical processing (“OLAP”), typically requires
 13 consideration or consolidation of several different tables or parts of tables, in order to organize
 14 “measures” — such as sales or expenses — according to multiple variables (or “dimensions”) —
 15 such as date, location and product type. (*Id.*) As a result, the data used in OLAP is often multi-
 16 dimensional. (*Id.*) To store and analyze such multi-dimensional data, a data structure called a
 17 “cube” or “OLAP cube” is used. (*Id.*) Although a cube is typically thought of as a three-
 18 dimensional object, in this context a cube can have fewer or more than three dimensions. (*Id.*)

19 2. **The Claims’ “Mapping” Is Stored**

20 The '321 patent relates to storing assignments of database tables (two-dimensional data
 21 structures) and/or OLAP cubes (multi-dimensional data structures) to classes. (Maier, ¶ 95.)
 22 Examples of classes in the '321 patent include “investment,” “equity,” or “goodwill.” (*See, e.g.*,
 23 Ex. 12, '321 Pat., at 2:20–23.) A stored assignment of tables or cubes to such classes allows a
 24 computer to retrieve tables or OLAP cubes assigned to a given class when performing actions
 25 related to that class. In some preferred embodiments, users select which tables or cubes to assign
 26 to a particular class. (*See, e.g., id.*, at 2:31–38; 4:29–32.) This selection (or “mapping,” used as a
 27
 28

verb) creates in storage an assignment or association (a “mapping,” used a noun) between the class and the selected tables/cubes.

In the ’321 patent the mapping is stored persistently so that it can be accessed and used by, for example, various application programs also running on the computer. (Maier, ¶ 95. *see, e.g.*, Ex. 12, ’321 Pat., at 2:12–15 (“For a given application program, a sub-set of the database tables may be assigned to the classes for access by the application program to the entities stored in the sub-set of these database tables. In one embodiment, the assignment of database table to classes can be stored in a mapping table.”) (addressing mapping of tables to classes); *see also id.* at 2:18–23 (addressing mapping of cubes to classes).) (*See* JCCS, at 59:15–25 (proposed FOF).)

In distinguishing the prior art Bedell reference, the inventor explained to the Examiner that storing a mapping for later use is a necessary part of the step of creating a mapping:

[O]nce a ‘custom grouping’ is made in Bedell there is no disclosure that the system retains any information concerning where each item in the ‘custom grouping’ originated. There is no explicit ‘mapping’ step disclosed in Bedell.

* * *

There can be no mapping in Bedell when the system contains no memory of what once was a part of one set and that has been moved to a different set. (Ex. 13, ’321 Pros. Hist., at pages SAP_02682958-59.) In other words, if a “mapping” is not stored, the system won’t work. SAP’s construction is true to this intrinsic evidence and states that a “mapping” must be stored.

Despite the intrinsic evidence to the contrary, Teradata declines to state that a “mapping” must be stored. Nevertheless, such a limitation is implicit in its proposed construction of “mapping table” as a “a computer-implemented data structure that holds associations or assignments.” This implicit recognition of the storage requirement is both inconsistent with Teradata’s proposal for “mapping” and insufficient to remedy the deficiency because “mapping table” appears in only three of the asserted claims.

3. The Claims’ “Mapping” Is Always Pairwise

In the ’321 patent, “mapping” assigns pairs of elements – either (a) a table and a class or (b) a cube and a class. For example, the ’321 patent states: “For instance, when the user selects

one of the OLAP cubes, the selected OLAP cube needs to be mapped to one of the predefined OLAP cube classes.” (Ex. 12, ’321 Pat., at 2:40–42.) Although multiple tables might be assigned to a given class, each individual assignment is made pairwise between the table and the class. This interpretation is consistent with “mapping” as understood by a POSITA. (Maier, ¶¶ 96–97.)

H. ’321 Patent - “Online Analytical Processing Cube”

SAP	Teradata
“a data structure or definition to store multidimensional data, where data to be stored in the data structure is or will be provided by online analytical processing”	“a data structure designed to store multidimensional data, where data to be stored in the data structure is provided by online analytical processing.”

Dispute: Does this term refer both to a physical instance of a data structure and also to the data-structure definition used to create that instance?

Some words refer both to a physical entity as well as to the definition of that entity, such as the word “route” referring both to a physical course and also to the step-by-step description of that course. The same is true of this claim term. It refers to an instantiated physical data structure in a computer’s memory. It also refers to descriptions stored in a computer that define that data structure and are used to create an instantiated example. (Maier, ¶¶ 101–102.)

The parties agree that an OLAP cube is a “data structure” to store multi-dimensional data. A commonly accepted definition of “data structure” has as its core element “an organizational scheme” applied to data. (Maier, ¶ 99, citing Ex. 14, MS Comp. Dict., at SAP_04709922.) Thus, “OLAP cube” is used to refer not only to a physical instance of a cube but also to such cube’s definition including its organizational scheme. (*Id.*, ¶¶ 101–102.) More generally, a POSITA would recognize that a computer implements descriptions that are provided to that computer. (*Id.*, ¶ 102.) In the context of OLAP processing, OLAP cubes are defined by descriptions that specify the characteristics of the cubes. (*Id.*) When a particular instance of a data structure is created, or “instantiated,” by consulting the relevant description, it has the properties that were defined for that data structure in the definition. (*Id.*) A POSITA would have understood that the terms “OLAP cube” and “data structure” often are used to refer both to the definition of a data

structure and a data structure instance that is created according to that definition. (*Id.*; JCCS 59:10–13 (proposed FOF).) SAP’s proposed construction makes this explicit. Teradata’s proposal does not.

The ’321 patent contains numerous statements indicating that OLAP cubes may be “pre-configured” or are “predefined,” before being instantiated. For example, the ’321 patent states that “[a] user may select from a set of predefined OLAP cubes.” (Ex. 12, ’321 Pat., at 2:17–18.) A POSITA would understand this statement as referring both to a set of definitions, and to data structures that can store data after being created and configured by implementing the definitions. (Maier, ¶ 101.) That an OLAP cube can mean a cube definition is further supported by the ’321 patent’s characterization that both queries and cubes can be “pre-configured objects.” A POSITA would understand a “query” to mean a set of instructions for retrieving particular data from a database system, as well as the process of executing those instructions. This understanding of “query” is analogous to an OLAP cube defining a layout for transactional data storage, as well as an object created according to that definition to hold the transactional data. (Maier, ¶ 101.)

I. ’321 Patent - “[Means For] Invoking An Online Analytical Processing Component To Fill The Online Analytical Processing Cubes With Transactional Data”

	SAP	Teradata
“[means for] invoking an online analytical processing component to fill the online analytical processing cubes with transactional data”	<p>“using software to transfer transactional data into online analytical processing cubes”</p> <p>This claim element is governed by 35 U.S.C. 112(6). <u>Structure/Material/Acts</u> An application program.</p>	<p>“using an online analytical processing component to transfer transactional data into empty online analytical processing cubes”</p> <p>This claim element is governed by 35 U.S.C. 112(6). <u>Corresponding Structure:</u> none; indefinite</p>

Disputes: Can only empty cubes be filled? Does the patent specification disclose structure for performing this function?

This term appears in the claims in two different contexts. First, it appears in method claims as a step to be performed. Second, it appears in apparatus claims as all or part of a

function of a “means-plus-function” element. In both contexts there is a dispute over the meaning of the term. In the second context there is also a dispute as to whether the specification adequately describes structure for performing the function.

1. There Is No Basis For Requiring That The OLAP Cube Be Empty

The parties disagree whether the “OLAP cubes” must be empty whenever transactional data is transferred into them.⁷ Nothing in the claims requires that the cubes be empty. Nothing in the specification *requires* this either, instead indicating that “[i]nitially, the OLAP cubes 328, 329, etc. may be empty and specify the layout for transaction data storage for various purposes.” (Ex. 12, ’321 Pat., at 4:66–5:1.) This language is expressly permissive (“may be”). No intrinsic evidence prohibits using this “fill” step in stages, repeatedly filling the cube with additional data. Further, both parties equate “fill” with “transfer” and one may, of course, “transfer” more material into a non-empty container.

2. The Specification Identifies An “Application” As The Associated Structure For Performing The “Invoking” Function

A means-plus-function claim element covers the “corresponding structure, material, or acts described in the specification and equivalents thereof.” 35 U.S.C. § 112(6). The specification must clearly link the corresponding structure, etc. to the recited function. *Medtronic, Inc. v. Advanced Cardiovascular Systems, Inc.* 248 F.3d 1303, 1311–13 (Fed. Cir. 2001). The specification-disclosed “structure,” etc. here is an application program: “In stage 408, the selected application program invokes an OLAP component in order to fill the selected OLAP cube with transactional data.” (Ex. 12, ’321 Pat., at 6:19–21.) This passage, and others, clearly links an “application program” with the recited “invoking” function.

Application programs were well known to the POSITA at the time the ’321 patent was filed, as indicated by the specification’s Background: “The SAP® business information

⁷ The proposed constructions also differ in that SAP specifies using “software” to perform the transfer while Teradata’s proposal specifies using “an online analytical processing component.” However, the parties have agreed that “online analytical processing component” means “software capable of processing online analytical processing cubes.” (ECF 206, (JCCS), p. 2.) Thus, this is not a substantive difference.

warehouse” is a data warehouse that includes “operative SAP applications” and “enables OLAP for processing of information.” Microsoft Excel is disclosed as another specific example of an application program that can be used to invoke OLAP functionality. (*Id.*, at 1:49–51.)

Application programs are further discussed in the Detailed Description of the ’321 patent, including how they can include programs that “perform a certain type of business-orientated data processing, such as for the purposes of accounting, book keeping and/or consolidation.” (*Id.*, at 3:45-51.)

The ’321 patent’s clearly-linked description that an application program is the “means for invoking” provides sufficient structure for this claim limitation. In *Zeroclick, LLC v. Apple Inc.*, in finding that claim terms did not trigger § 112(6), the Federal Circuit held that the use of terms such as “program” and “user interface code” were “used not as generic terms or black box recitations of structure or abstractions, but rather as specific references to conventional [structures] existing in the prior art at the time of the inventions.” 891 F.3d 1003, 1008 (Fed. Cir. 2018). The rationale applied in *Zeroclick* applies with equal force in concluding that “application programs” is sufficient disclosed structure for this means-plus-function claim term, particularly because the specification identifies particular examples of relevant types of application programs.

J. ’516 Patent - “Eviction Policy Plug-In” / “Storage Policy Plug-In” / “Plug-In”

	SAP	Teradata
eviction policy plug-in	“the actual piece of software or code that dictates the removal of an object from cache”	“plug-in for performing a sorting method and an eviction timing method”. See above for “plug-in” and supra § A.5 for “sorting method” and “eviction timing method.”
storage policy plug-in (or storage plug-in) ⁸	“the actual piece of software or code that executes the “get” and “put” operations for objects stored in cache”	“plug-in for performing cache storage treatment of stored data.” See above for “plug-in”.

⁸ The terms “storage policy plug-in” and “storage plug-in” are used interchangeably in the ’516 patent. (*See, e.g.*, Ex. 15, ’516 Pat., at 7:37–40 (“Another or related storage policy plug-in function may be used to perform a “write-through” process, in which a “put” of an object into

1 2 3 4 5	plug-in	“piece of software or code that provides a requisite functionality”	“a discrete body of code that is separate from a main program and can be added to and removed from the main program without modification to the main program, such that the discrete body of code adds functionalities to the main program when added to it”
-----------------------	---------	---	--

6 Dispute: Should “eviction policy plug-in” and “storage policy plug in” each be construed
7 in the context of the patent’s intrinsic evidence for that coined term, rather than selectively
8 picking pieces of extrinsic evidence for the term “plug-in” alone, which has several different
9 meanings?

10 1. **’516 Patent Background**

11 The ’516 patent relates to caching technologies. (Maier, ¶ 55.) Caching increases
12 computer system performance because information within cache memory can be accessed faster
13 as compared to disk or other external memory. (*Id.*) However, cache memory is typically much
14 smaller than this other storage. (*Id.*) Accordingly, a goal of caching is to store commonly used
15 information in the cache while avoiding less commonly used data. (*Id.*) This is roughly akin to
16 having a few books on a nightstand with the rest on bookshelves.

17 One important function of cache management is determining which objects in cache
18 should be removed (or “evicted”) from cache to make room for new incoming objects. Another
19 is getting and putting objects stored in the cache. (*Id.*, at ¶ 56.) The ’516 patent describes that
20 these two functions are performed by the “eviction policy plug-in” and “storage policy plug-in,”
21 respectively.

22
23
24
25
26
27 _____
28 cache automatically results in a copy of that object being directed to storage space 440.”); 7:1–4
 (“The storage plug-in may be, in one embodiment, the actual piece of software or code that
 executes the ‘get’ and ‘put’ operations for the objects stored according to the treatment
 determined by the associated cache region.”).)

1 **2. The Context Of The '516 Patent**
2 **Controls The Meaning Of The "Plug-In Terms"**

3 “Eviction policy plug-in” and a “storage policy plug-in” are not terms of art and have no
4 special meaning in the art. (Maier, ¶¶ 105, 107; JCCS, at 65:11–19 (proposed FOF).) The patent
5 clearly states what the terms mean within that context. (Maier, ¶¶ 106, 108.)

6 The patent states: “The eviction policy plug-in may be, in one embodiment, the actual
7 piece of software or code that dictates the removal of an object from cache (e.g., when some
8 form of cache capacity threshold is exceeded).” (Ex. 15, '516 Pat., at 7:8–11; Maier, ¶ 106.)
9 Similarly, “The storage plug-in may be, in one embodiment, the actual piece of software or code
10 that executes the ‘get’ and ‘put’ operations for the objects stored according to the treatment
11 determined by the associated cache region.” (Ex. 15, '516 Pat., at 7:1–4; Maier, ¶ 108.) While
12 these statements refer to “one embodiment,” there is no contrary embodiment or different
13 definition in the patent. SAP’s proposed constructions quote verbatim from these definitions in
14 the specification.

15 In contrast, Teradata’s proposed constructions are divorced from the intrinsic evidence in
16 favor of selective reliance on pieces of extrinsic evidence. This avoidance of intrinsic evidence is
17 evidenced by Teradata’s efforts to construe the term “plug-in” alone and in isolation, rather than
18 construe the full terms “eviction policy plug-in” and a “storage policy plug-in” as actually used
19 in the '516 patent. SAP objects to construing the term “plug-in” in isolation as it risks jury
20 confusion and leaves out the context specified by the patent.

21 Teradata’s primary reliance on extrinsic evidence is particularly inappropriate here, for
22 two reasons. First, that evidence shows the general term “plug-in” to have several different
23 meanings in the art. (Maier, ¶ 109). For example, one dictionary Teradata cites states that “plug-
24 ins have to be loaded at the same time as the main program; they then show up as an option in an
25 appropriate menu” (Ex. 16, Dict. of Computer and Internet Terms, p. TD01892045), but these
26 requirements are absent from Teradata’s other extrinsic evidence. (Maier, ¶ 109.)

27 Second, Teradata selectively picks aspects from its extrinsic evidence’s disparate uses of
28 “plug-in” for its proposed construction, even when those aspects conflict with the '516 patent’s

1 disclosures. For example, Teradata’s proposed construction requires that a plug in “adds
2 functionalities to the main program when added to it.” Presumably this requirement is based on
3 Teradata’s citation to two dictionary definitions. (*See* JCCS Ex 5, at 20 (citing Ex. 16, Dict. of
4 Computer and Internet Terms, p. TD01892045; Ex., 17, Modern Dict. of Elects., p.
5 TD1893775).) However, the cache manager of the patent will not operate without the “eviction
6 policy plug-in” and a “storage policy plug-in.” (Maier, ¶ 110.) Thus, in the context of the ’516
7 patent the eviction policy plug-in and storage policy plug-in provide *required* functionality, not
8 “additional functionality.” This contrasts the claims’ two plug-ins from some different plug-ins
9 in other environments, like an internet browser or an e-mail application, where the plug-in is an
10 “accessory program that provides additional functions.” (*See, e.g.*, Teradata’s citation to Ex. 16,
11 Dict. of Computer and Internet Terms, p. TD01892045.) Thus, while Teradata’s insistence that
12 plug-ins are “separate from a main program,” and dynamically “added and removed without
13 modification to the main program,” may apply to some plug-ins in some environments, it does
14 not apply to the claimed plug-ins in the ’516 patent.

15 As further examples of Teradata cherry picking from the extrinsic evidence, the Barron’s
16 dictionary definition states that when loaded, “[plug-ins] show up as an option in an appropriate
17 menu.” However, this is not part of Teradata’s proposed construction. Similarly, Decasper states
18 that “plugins are code modules which implement a specific EISR functionality” and that “[t]his
19 architecture allows code modules, called *plugins* to be dynamically added and configured at run
20 time.” (*See* Ex. 18, Decasper, pp. TD01660464, TD01660466.) Neither of these features is
21 reflected in Teradata’s construction.

22 Teradata’s proposed constructions also include limitations that are not supported in either
23 the cited intrinsic evidence or the cited extrinsic evidence. For example, none of the cited
24 evidence indicates that a “plug-in” must be “a discrete body of code,” or requires that a “plug-in”
25 “can be added to and removed from the main program without modification to the main
26 program.”
27
28

1 **IV. CONCLUSION**

2 For the reasons set forth above, the Court should adopt SAP's proposed constructions for
3 all the disputed terms.

4 Dated: February 13, 2020

Respectfully submitted,

5 KLARQUIST SPARKMAN, LLP

6 By: /s/ John D. Vandenberg

7 John D. Vandenberg

8 Counsel for Defendants

9 SAP SE, SAP AMERICA, INC., AND SAP LABS, LLC